Distributed and Scalable Anisomorphic Geometric Simplification

Category: Research/Systems; Format: Spoof

Abstract

Geometric simplification is a well-known technique for simplifying complex geometric models. Previous work in this area in the past has consisted primarily of lossless compression and error-bounded simplification techniques. By removing such constraints, we can *anisomorphically* reduce a model's representation—cheaply and efficiently—down to as little as 0% of its original size. We perform such simplification in two stages, which can be employed independently or combined for maximum effect. The first stage, the *photoreplication* stage, uses only items commonly found in the home or office, and is highly scalable. The second stage (called the *first* stage for historical reasons) takes advantage of idle distributed resources to further simplify a model. We briefly describe our implementation, and emphasize the ease with which it can be integrated into existing systems.

Keywords: geometric simplification, distributed processing, triangulation, buzzwords, photoreplication, anisomorphic compression

1 Introduction

Geometric simplification is a well-known technique for simplifying complex geometric models. Previous work in this area has consisted of methods which bound the error between the original and simplified model [3], as well as those that implement lossless or error-bounded compression [1].

Our research shows that removing such relational constraints between the original and simplified model generates many new possibilities for simplification. We name this class of functions *anisomorphic* simplification, as they need not preserve the original topology. Experiments have shown that two such techniques stand out as being efficient in computation, simple to implement, and enjoyable to watch being employed; and further, that the use of these methods in successive stages generates excellent pictures.

2 Photoreplication Stage

In the photoreplication stage, we anisomorphically simplify the model through the use of an input-output feedback loop. The point set for the model is printed to paper, then photocopied n times with a paper duplicator, and then input back to digital format with what we call a scan-converter. If the number of points in the original model is P_0, and the loss-factor of printing, duplicating, and scan-conversion are P_1, P_2, and P_3 respectively, then the number of points in the simplified model is

$$P' = P_0 \times P_1 \times P_2^n \times P_3$$

The simplification is easily scalable, then, by the purchase of additional duplication devices. It is in fact possible to simulate a n-tier photoreplication process with a single duplicator; we state without proof [2] that this produces results within human visual tolerance. The method does necessitate a random image rotation at each stage, to reduce the development of duplicator artifacts.

The photoreplication stage is necessarily limited by the dimensionality of the print medium, at present 2; the new model must be generated from the points through one of several popular over-the-counter triangulation methods [4]. Later in this paper we discuss planned extensions to overcome this dimensionality problem.

3 First Stage

The first stage anisomorphically simplifies by use of distributed computing resources, taking advantage of known wide-area network properties such as congestion and packet loss [5]. The model is split into lowest-level primitives, such as points or triangles, and each primitive is wrapped with a unique identifier in a separate unreliable UDP packet. The packets are routed through each of the available workstations in turn, while network congestion eliminates some random subset of primitives. The host machine then reconstructs the simplified model from the reduced subset, using the unique identifers to facilitate proper reordering.

The loss-level of the network is the most important factor in determining loss in this stage, to wit:

$$P' = P_0 \times \lambda$$

where λ is the network lossiness factor. The lossiness is determined by network congestion, individual machine loads, sysadmin whim, temperature, and whether anyone has tripped over the ethernet cables recently.

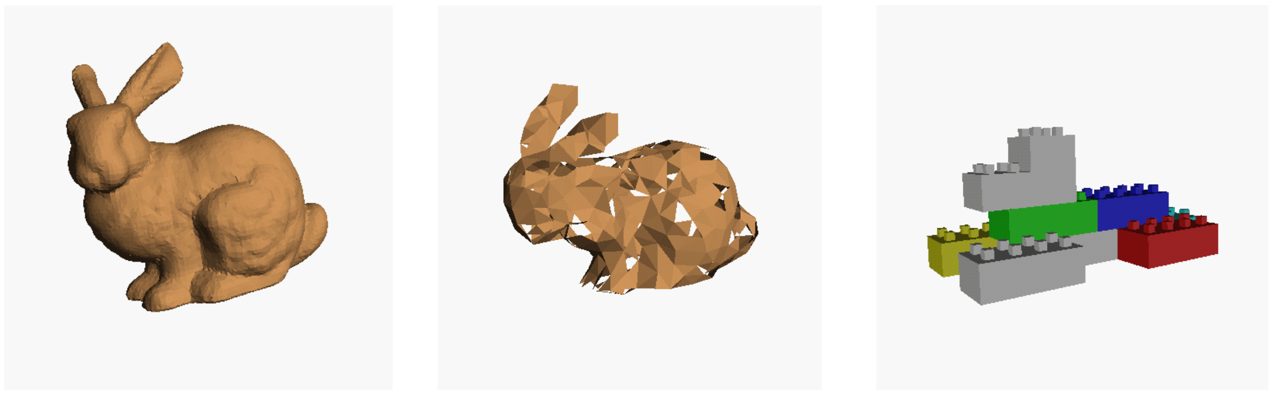

Figure 1: Bunny model: original, simplification stage 1, and simplification stage two

4 Implementation and Results

We were surprised, for the *photoreplication* stage, how particularly acute were the idiosyncracies of each duplicator and scan-converter. Extensive testing, via photoduplication of corporeal models, showed that some duplicators were unsuitable for our simplification method. Though our research assistants preferred the self-collating model, we eventually settled upon a duplicator whose fingerprint-marred glass most efficiently simplified the model. For scan-conversion, we employed a popular hand-held half-page device that gave the highest probability of error. For all of the color images, we simulated a 10-tier photoreplication process with a single duplicator and scan-converter, employing the method discussed in Section 2.

In our original implementation, the *first* stage used a cluster of twenty SGI O2 workstations, each with R10000 processors and 512MB of memory, connected by a 100 Mbit ethernet network. We introduced random loss by inviting local high-school students to use our machines in a multi-player Quake[1] tournament, thereby overloading the local network. Unfortunately, we were unable to duplicate these conditions for time trials as other laboratory projects took "priority" when this deadline approached. We simulated our previous experiments with four 486-DX PCs, a Linux box, and two Palm Pilots[2] connected to the Linux machine via Pilot modems.

Figure 4 shows the combined effect of the photoreplication and first stages, which took approximately two hours to compute. The author was fortunate to win the first match, 20 to 14, and lost the remaining three to worthy opponents. The table in Figure 3 shows the number of points in each model, given each combination of simplification methods.

5 Conclusion and Future Work

We have demonstrated and implemented two distributed and hierarchical geometric simplification processes, which actually see improved performance under severe network and processor load. These algorithms are easy to implement or integrate into existing code base; in fact, developing a full system only took a few hours, including the design and fabrication of special-purpose photoreplication hardware.

We look forward to some improvements in the current system. For instance, the photoreplication

[1]Quake is a trademark of IdSoftware
[2]Palm Pilot is a trademark of 3Com

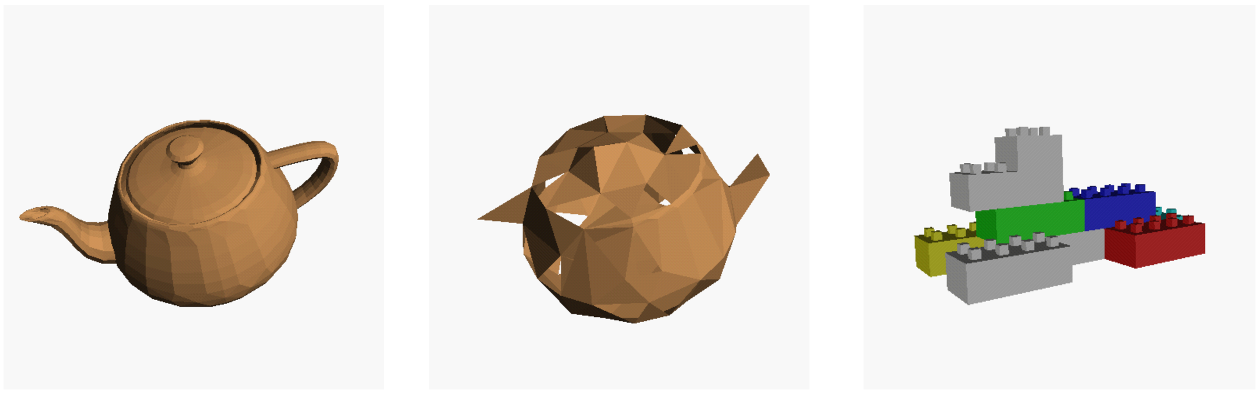

Figure 2: Teapot model: original, simplification stage 1, and simplification stage 2. Original is on the left.

simplification method	bunny model	teapot model	cube model
none	8145	1178	8
photoreplication	2142	352	4
distributed	6332	953	5
combination	674	174	1

Figure 3: Simplification results: each entry is number of remaining points.

stage currently projects 3-D point sets into two dimensions. We are working on a method to choose the projections properly to facilitate reconstruction of the original three-dimensional position of points remaining in the simplified version. We hope to incorporate color into the entire photoreplication stage, though we hesitate to postulate about the effects of such a major step.

References

[1] Fowler, J., and Yagel, R. Lossless compression of volume data. In *1994 Symposium on Volume Visualization* (Oct. 1994), A. Kaufman and W. Krueger, Eds., ACM SIGGRAPH, pp. 43–50. ISBN 0-89791-741-3.

[2] Heckbert, P. S. Ray tracing Jell-O brand gelatin. In *Computer Graphics (SIGGRAPH '87 Proceedings)* (July 1987), M. C. Stone, Ed., vol. 21, pp. 73–74.

[3] Kalvin, A. D., and Taylor, R. H. Superfaces: Polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Applications 16*, 3 (May 1996), 64–77. ISSN 0272-1716.

[4] Mucke, E. *Shapes and Implementations in Three-Dimensional Geometry.* PhD thesis, University of Illinois at Urbana-Champaign, 1993.

[5] Peterson, L., and Davie, B. *Computer Networks.* Morgan Kaufman, 1996.